**OBE IMPLEMENTATION: SCHOOL SETTING**

*by*

**M**.**DEEPTHI [AP22110010275]**

**M.RADHA KALYANI [AP22110010244]**

**J.RAMYA REDDY[AP22110010209]**

*A report for the CS307:Mobile Application Development using JAVA*



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SRM UNIVERSITY AP:: AMARAVATI**

**[April,2025]**

**INDEX**

**Introduction 3**

Project Modules: 4

**Architecture Diagram 5**

**Module Description 6**

Programming Details naming conventions to be used: 7

Table details: Schools 7

**Source Code 8**

**Screen Shots 18**

**Conclusion 23**

**INTRODUCTION**

Outcome-Based Education (OBE) is a student-centric teaching and learning model that focuses on measuring student performance through outcomes. It emphasizes the achievement of specific competencies at the end of educational experiences. SRM University - Andhra Pradesh has adopted the OBE framework across its academic infrastructure to align its curriculum and evaluation system with international standards.

The purpose of this document is to present a detailed project report on the development of the "Schools" module, which is a part of the comprehensive OBE Implementation System. The Schools module is one of the foundational layers of the system, enabling the management of various schools under the university's administration. Developed using Java and integrated with a MySQL database, this module supports Create, Update, Retrieve, and Delete (CURD) operations for school entities.

**Project Modules**

The complete OBE system comprises multiple modules, each handling a distinct component of the academic and outcome assessment process. These modules are:

1. Blooms Level Setting
2. Program Level Objective Setting
3. University
4. **Schools (Current Module)**
5. Departments
6. Programs
7. Courses
8. Course Objective Setting
9. Course Outcome Setting
10. Course Articulation Matrix Setting
11. Course Utilization Setting
12. Course Reference Setting

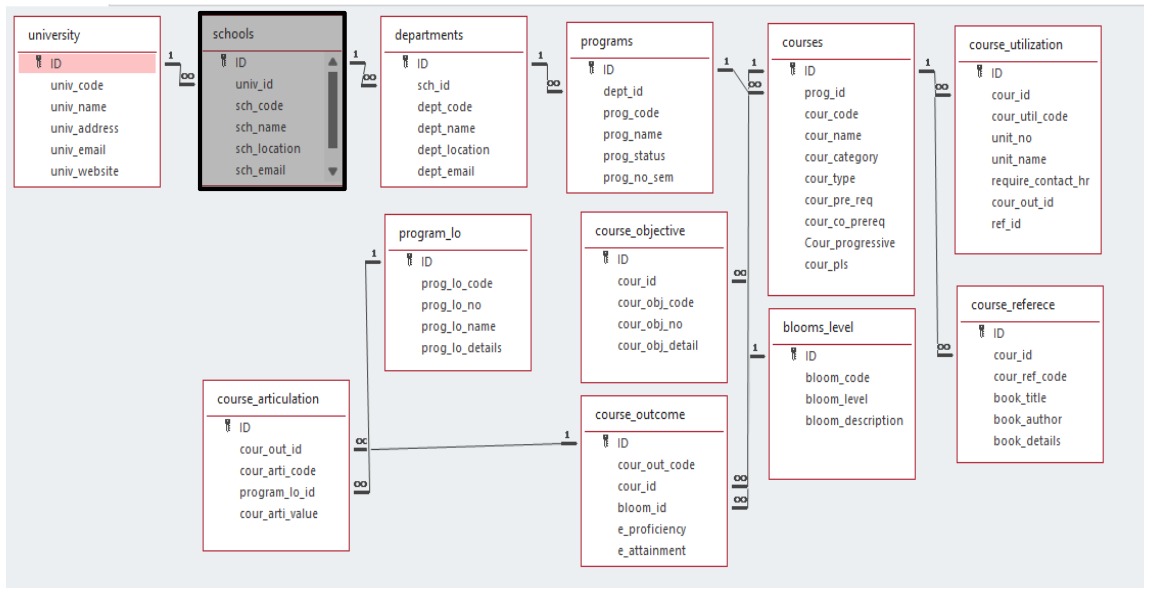
Each of these modules works in tandem with one another to deliver a cohesive academic management experience. The Schools module, in particular, plays a crucial role in managing the academic structure by facilitating the administration of different schools within SRM-AP.

**ARCHITECTURE DIAGRAM**

The architecture of the system follows a modular and layered design pattern. Each module is loosely coupled with others, which allows for independent development and deployment. The Schools module interacts primarily with the University module and serves as a prerequisite for the Departments module.

* **Frontend**: Java Swing (AWT)
* **Backend**: JDBC for database connectivity
* **Database**: MySQL

**Key features of the architecture:**

* User interface built with Java Swing
* CRUD operations handled via event-driven programming
* Data persisted in MySQL with standard queries
* Reusable components for uniform design

**Module Description**

**Module Name:** Schools

**Objective**: The primary goal of the Schools module is to enable university staff to **create**, **retrieve**, **update**, and **delete** (CRUD) information associated with schools under SRM-AP. This includes storing metadata such as the school code, name, location, email, and its link to the university entity via a unique university ID (uni\_id).

**Functionalities:**

* Add new school details
* Update existing school information
* Delete school records
* Retrieve and display school details based on school code

**Inter- Module Dependency:**

* Depends on University module (for uni\_id linkage)
* Provides data to Departments module (for sch\_code reference)

**Target Users:**

* University administrators
* Academic coordinators
* IT system administrators

**Programming Details**

The Schools module has been implemented using Java programming language. The graphical user interface (GUI) is built using Java AWT (Abstract Window Toolkit), providing a clean and functional layout for user interaction.

Naming Conventions Used:

* Class/Activity: schools\_Schools
* Functions:
  + Create: create.schools()
  + Update: update.schools()
  + Retrieve: retrieve.schools()
  + Delete: delete.schools()

These functions handle the corresponding CRUD operations via button clicks in the Java Swing application.

**Table Details: schools**

The module uses a MySQL database with the following schema for the "schools" table:

| **Field Name** | **Data Type** |
| --- | --- |
| id | Integer |
| uni\_id | String |
| sch\_code | String |
| sch\_name | String |
| sch\_location | String |
| sch\_email | String |

**SOURCE CODE**

package schools;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Schoolss extends Frame {

Label l1, l2, l3, l4, l5;

TextField tfUniID, tfSchCode, tfSchName, tfSchLocation, tfSchEmail;

Button btnAdd, btnUpdate, btnDelete, btnRetrieve;

Connection con;

Schoolss() {

setTitle("University School Management");

// Labels

l1 = new Label("University ID:");

l2 = new Label("School Code:");

l3 = new Label("School Name:");

l4 = new Label("School Location:");

l5 = new Label("School Email:");

// TextFields

tfUniID = new TextField();

tfSchCode = new TextField();

tfSchName = new TextField();

tfSchLocation = new TextField();

tfSchEmail = new TextField();

// Buttons

btnAdd = new Button("Add");

btnUpdate = new Button("Update");

btnDelete = new Button("Delete");

btnRetrieve = new Button("Retrieve");

// Layout

setLayout(null);

int xLabel = 50, xField = 200, width = 200, height = 25;

l1.setBounds(xLabel, 50, 120, height); tfUniID.setBounds(xField, 50, width, height);

l2.setBounds(xLabel, 90, 120, height); tfSchCode.setBounds(xField, 90, width, height);

l3.setBounds(xLabel, 130, 120, height); tfSchName.setBounds(xField, 130, width, height);

l4.setBounds(xLabel, 170, 120, height); tfSchLocation.setBounds(xField, 170, width, height);

l5.setBounds(xLabel, 210, 120, height); tfSchEmail.setBounds(xField, 210, width, height);

btnAdd.setBounds(50, 260, 100, 30);

btnUpdate.setBounds(160, 260, 100, 30);

btnDelete.setBounds(270, 260, 100, 30);

btnRetrieve.setBounds(380, 260, 100, 30);

add(l1); add(tfUniID);

add(l2); add(tfSchCode);

add(l3); add(tfSchName);

add(l4); add(tfSchLocation);

add(l5); add(tfSchEmail);

add(btnAdd); add(btnUpdate); add(btnDelete); add(btnRetrieve);

connectToDatabase();

btnAdd.addActionListener(e -> addSchool());

btnUpdate.addActionListener(e -> updateSchool());

btnDelete.addActionListener(e -> deleteSchool());

btnRetrieve.addActionListener(e -> retrieveSchool());

setSize(550, 350);

setVisible(true);

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

try {

if (con != null) con.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

dispose();

}

});

}

private void connectToDatabase() {

String url = "jdbc:mysql://localhost:3306/schoolss\_db?useSSL=false";

String user = "root";

String password = "radha@08"; // Masked for security

try {

Class.forName("com.mysql.cj.jdbc.Driver");

con = DriverManager.getConnection(url, user, password);

System.out.println("Connected to MySQL database successfully!");

} catch (Exception e) {

e.printStackTrace();

showMessage("Database Connection Failed!");

}

}

private boolean fieldsNotEmpty() {

return !tfUniID.getText().trim().isEmpty() &&

!tfSchCode.getText().trim().isEmpty() &&

!tfSchName.getText().trim().isEmpty() &&

!tfSchLocation.getText().trim().isEmpty() &&

!tfSchEmail.getText().trim().isEmpty();

}

private void showMessage(String message) {

Dialog d = new Dialog(this, "Message", true);

d.setLayout(new FlowLayout());

d.setSize(300, 100);

d.add(new Label(message));

Button ok = new Button("OK");

ok.addActionListener(e -> d.setVisible(false));

d.add(ok);

d.setLocationRelativeTo(this);

d.setVisible(true);

}

private void clearFields() {

tfUniID.setText("");

tfSchCode.setText("");

tfSchName.setText("");

tfSchLocation.setText("");

tfSchEmail.setText("");

}

private void addSchool() {

if (!fieldsNotEmpty()) {

showMessage("Please fill all fields.");

return;

}

String query = "INSERT INTO schools (uni\_id, sch\_code, sch\_name, sch\_location, sch\_email) VALUES (?, ?, ?, ?, ?)";

try (PreparedStatement pst = con.prepareStatement(query)) {

pst.setString(1, tfUniID.getText());

pst.setString(2, tfSchCode.getText());

pst.setString(3, tfSchName.getText());

pst.setString(4, tfSchLocation.getText());

pst.setString(5, tfSchEmail.getText());

pst.executeUpdate();

showMessage("School Added Successfully!");

clearFields();

} catch (SQLException e) {

e.printStackTrace();

showMessage("Failed to add school!");

}

}

private void updateSchool() {

if (!fieldsNotEmpty()) {

showMessage("Please fill all fields.");

return;

}

String query = "UPDATE schools SET uni\_id = ?, sch\_name = ?, sch\_location = ?, sch\_email = ? WHERE sch\_code = ?";

try (PreparedStatement pst = con.prepareStatement(query)) {

pst.setString(1, tfUniID.getText());

pst.setString(2, tfSchName.getText());

pst.setString(3, tfSchLocation.getText());

pst.setString(4, tfSchEmail.getText());

pst.setString(5, tfSchCode.getText());

int rows = pst.executeUpdate();

if (rows > 0) {

showMessage("School Updated Successfully!");

} else {

showMessage("No School Found to Update!");

}

} catch (SQLException e) {

e.printStackTrace();

showMessage("Update Failed!");

}

}

private void deleteSchool() {

if (tfSchCode.getText().isEmpty()) {

showMessage("Enter School Code to delete.");

return;

}

String query = "DELETE FROM schools WHERE sch\_code = ?";

try (PreparedStatement pst = con.prepareStatement(query)) {

pst.setString(1, tfSchCode.getText());

int rows = pst.executeUpdate();

if (rows > 0) {

showMessage("School Deleted Successfully!");

clearFields();

} else {

showMessage("No School Found to Delete!");

}

} catch (SQLException e) {

e.printStackTrace();

showMessage("Deletion Failed!");

}

}

private void retrieveSchool() {

if (tfSchCode.getText().isEmpty()) {

showMessage("Enter School Code to retrieve.");

return;

}

String query = "SELECT \* FROM schools WHERE sch\_code = ?";

try (PreparedStatement pst = con.prepareStatement(query)) {

pst.setString(1, tfSchCode.getText());

try (ResultSet rs = pst.executeQuery()) {

if (rs.next()) {

tfUniID.setText(rs.getString("uni\_id"));

tfSchName.setText(rs.getString("sch\_name"));

tfSchLocation.setText(rs.getString("sch\_location"));

tfSchEmail.setText(rs.getString("sch\_email"));

showMessage("School Retrieved Successfully!");

} else {

showMessage("No School Found!");

}

}

} catch (SQLException e) {

e.printStackTrace();

showMessage("Retrieve Failed!");

}

}

public static void main(String[] args) {

new Schoolss();

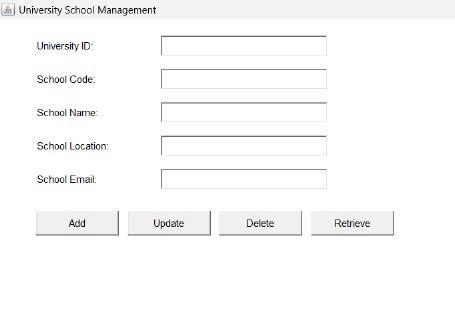
}

}

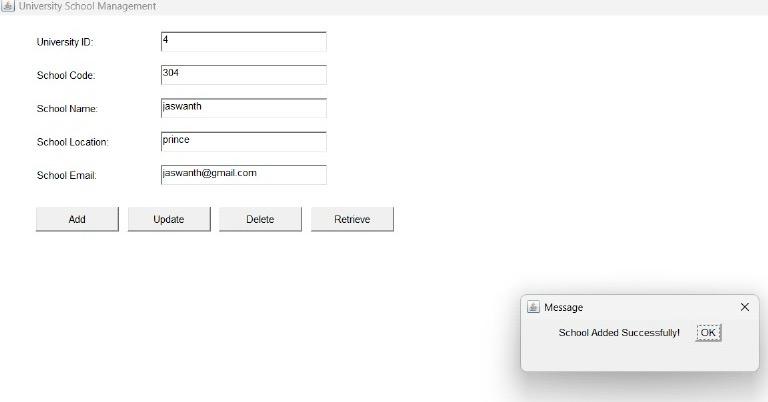
**OUTPUT**

**Screen Shots :**

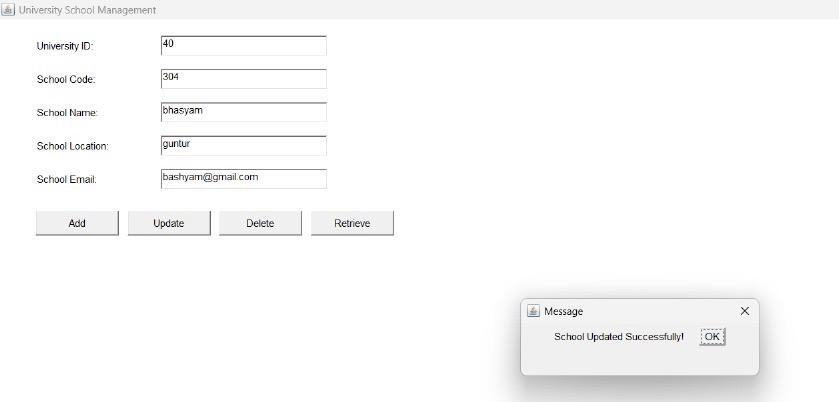
**FORM:**

****

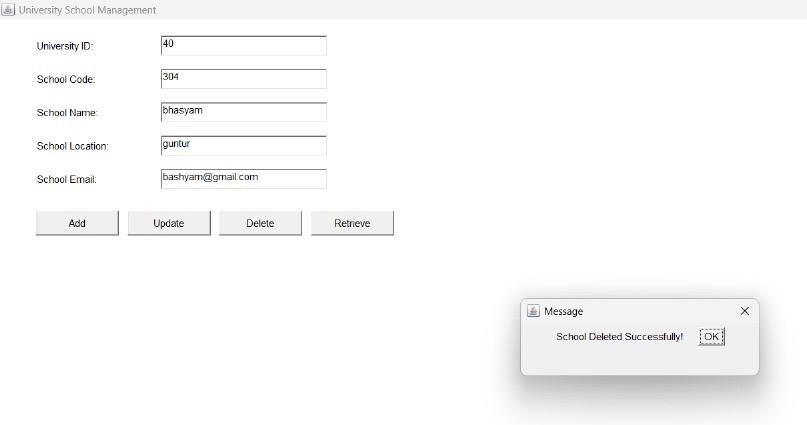
**ADD:**

****

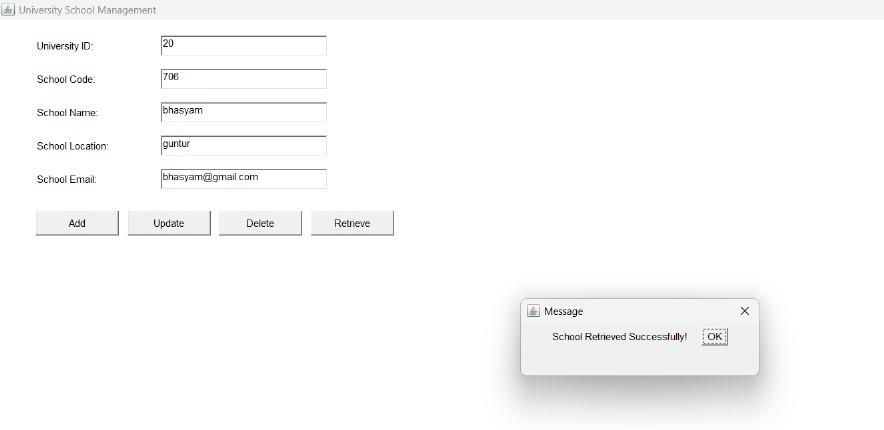
**UPDATE:**

****

**DELETE:**

****

**RETRIEVE :**

****

**Conclusion**

The Schools module plays an integral role in the OBE Implementation System at SRM-AP. It provides university administrators with the ability to manage core academic entities in a structured and secure way. By leveraging Java's capabilities for desktop application development and MySQL's robustness for data storage, the module ensures reliability, usability, and scalability.

**Future enhancements may include:**

* Role-based access control
* Enhanced UI/UX using JavaFX
* RESTful APIs for web integration
* Validation enhancements for field inputs

In summary, the Schools module not only simplifies school management but also acts as a foundation for the broader academic structure under the OBE framework. The successful implementation of this module paves the way for a smooth rollout of additional modules such as Departments, Programs, and Course Management.

This module has been developed with modularity, reusability, and future scalability in mind, aligning well with the long-term goals of the SRM-AP academic ecosystem.